An RTM based Distributed Simulation System for Guide Robot

Chen Peihua, Cao Qixin, Yang Yang and Leng Chuntao
Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai, China, 200240

Abstract: In order to enhance the robot system integration and development for guide robot, a distributed simulation system was developed in this study using RTM (Robot Technology Middleware) technology, which is an open software platform for robot systems. The RT (robot technology) system of an adapter, a controller and the robot, together with other CORBA objects, was developed to connect the graphical programming interface with 3D simulator to set up an RTM based distributed simulation system. Simultaneously, the application of the distributed simulation system also confirms the controlling of the real robot utilizing the RT system. The proposed distributed simulation system based on RTM can obviously accelerate the software component development as well as the system integration for guide robot, which will certainly lower the cost of the development of new robot application systems.

Keywords: Distributed simulation system, graphical programming, RT component, RTM

INTRODUCTION

Recently, with the development of the Internet and communication technologies, the quantity of research and development projects which aim at making robots and robot systems more intelligent by distributing their necessary resources over a network, is increasing (Kim et al., 2004; Quintas et al., 2011; Zhang et al., 2009b). In the near future, human beings will live in a world where all objects such as electronic appliances are networked to each other and robots will provide us with various services by any device through network as we need (Kim, 2005). However, as the level of structural and behavioral complexity increases in robot systems, the need for technologies supporting the software module development and the integration of those systems has expanded (Mohamed et al., 2008). To the component development, a systematic methodology and infrastructure for the local and distributed composition of modular functional components is needed. And for the distributed simulation system integration, robot middleware technology enables the development of robot systems from the traditional close mode to an open, comprehensive and integrated mode. The standardization of components and a lot of underlying service will achieve the rapid development and maintenance of robot systems.

A distributed simulation system has been developed by our URPE (Ubiquitous Robotics and related Programing Environment) project since 2005 (Chen et al., 2009; Qiu et al., 2005). At the same time, we also developed hardware models, key technologies and implemented CORBA (Common Object Request Broker Architecture) based servers for the robot systems (Zhang et al., 2009a). In this study, in order to solve the increasing functional component development and the system integration of guide robot and take full advantages of the network and computer technologies, we use the RTM (Robot Technology Middleware) technology (Ando et al., 2005) to set up several key modular software components and build a distributed simulation system for guide robot. The distributed simulation system can help the developers improve the efficiency of functional component and application development.

ROBOT TECHNOLOGY MIDDLEWARE

RTM: Robot Technology Middleware (RTM) is developed in collaboration between the Japanese Ministry of Economy, Trade and Industry (METI), the Japan Robot Association (JARA) and National Institute of Advanced Industrial Science Technology (AIST), to promote the application of RT in various fields. To make robot systems be language and platform independent, RTM has been developed based on CORBA using a number of specifications at the distributed middleware interface level. A prototype implementation named Open RTM-aist (http://www.openrtm.org/) is released (Ando et al., 2008). The main goals of RTM are to build robots and their functional parts in a modular structure at the software
level as well as to simplify the process of building robots by combining selected modules simply, which allow system designers or integrators to build customized robots for a variety of applications in an effective and efficient manner.

The RT-Middleware would support the construction of various networked robotic systems by the integration of various network enabled robotic elements, which are called RT-Components.

**RT component:** Towards robotic system development and modular design features, Japan has proposed the Robot Technology Component (RTC) (Ando *et al.*, 2006), which is based on the CORBA specification and has made some simplifications and complementation. Using RT-Middleware to integrate different RTCs will speed up the robot system components integration from the module level.

RTC is the basic unit in a modular robot system. It is used for functional encapsulation of different hardware and software resources and provides CORBA-compliant RTC interfaces for the external and realizes plug and play system integration through the CORBA bus. Figure 1 shows the architecture block diagram of the RT Component.

**SYSTEM ARCHITECTURE**

For the purpose of making modular software components’ development faster and robot system integration easier, we developed the distributed simulation system as shown in Fig. 2.

**Graphical Programming Interface** is a programming tool used to specify the robot’s tasks by using a list of
motion icons (Fig. 3). Once an icon is programmed, it

can be saved into an icon list. With this tool, the user

can program the robot to complete multiple motion
tasks. As a user-programming tool, it provides a user

interface to send commands to the related units of the

robot or even a networked sensor and integrates them

with the user’s commands to the RTCs via Unit

Emulator.

Unit Emulator is a motion engine that can power

on/off, servo on/off the robot units (such as Mobile

Unit, Right Arm Unit, Left Arm Unit, etc.) and translate

the commands from the graphical programing interface
to the set Pos command. UnitEmulator is implemented

as a CORBA object, as presented in Fig. 2. In a

UnitEmulator server, we need to set the configuration
of Unit Emulator, as shown in Table 1. In Table 1, we

set the “NameService IP” to “localhost”, “NameService

Port” to “2809”, “Unit Name” to “MnpL” (means left

arm here), “Viewer Name” to “Sim3D.object” and

“Segment Period” to “100”.

The RT Component of Adapter is developed to

receive the commands and data from the UnitEmulator

and transform them to a right format or order which the

Controller RTC can use. In order to connect with

UnitEmulator, we create CORBA interfaces which are

defined in a IDL (Interface Definition Language) file

(Aleksy et al., 2005).

In Controller RTC, some algorithms like path

planning, motion planning can be implemented through

using the data sent by the Adapter RTC and the sensors’

feedback, furthermore, the results will be output to the

VirtualRobot RTC in 3D Viewer part. The

VirtualRobot RTC is set up by utilizing the Controller

Bridge tool with the controller Bridge parameters in

OpenHRP (Kanehiro et al., 2004) and connected to the

Controller RTC at the same time.

In 3D Viewer part, there are other CORBA server

objects, which could load in the robot and the

environment as VRML models, display the simulation,

calculate the collision detection, etc.

By using the RTM technology, we can focus on the

developments of the modular components’ functions,

distribute them around a networked environment and

integrate them as an RT system easily.

APPLICATION OF THE DISTRIBUTED

SIMULATION SYSTEM

The distributed simulation system has been

applied to a Guide Robot named Jiao Long Service

Robot (JLSR) (Fig. 4). This robot is developed by

Research Institute of Robotics, Shanghai Jiao Tong
Fig. 4: JLSR robot and degree of freedom configuration

Fig. 5: JLSR robot welcomes a guest and picks up a cup of tea

Fig. 6: Structures of JLSRA adapter and JLSR controller RT components
University, for a robot restaurant. It has four omnidirectional wheels in the base, 6 DOFs (degrees of freedom) in one arm, one DOF in one gripper and one DOF in the head part. Take an application of this robot for example, when a guest comes, the JLSR will make a warm welcome firstly, then introduce the restaurant and pick up a cup of tea to the guest, as interpreted in Fig. 5. The two pictures in Fig. 5 illustrate the simulation of the JLSR robot that welcomes a guest (here we use another robot instead) at the door and picks up a bottle from a table.

In this application, there are three kinds of RT Components, which are JLSRA dapter, JLSR controller, JLSR robot. The structures of JLSRA dapter RTC and JLSR controller RTC can be seen in Fig. 6.

The JLSRA dapter RTC receives the command of setPos from UnitEmulator COBAR servers, which includes the unit name and its reference positions a current time and transforms them to the transformation matrix of the vehicle unit and the angles of all joints, after which it writes the data to its OutPort (transfseqo and valueseqo respectively). The flow diagram of the above procedure is shown in Fig. 7. In Fig. 7, after JLSRA dapter RTC is activated, it will run the execution context periodically. In one loop of the execution, it first check the value of “Cmd”, it will write the data to “transfseqo” OutPort if “Cmd” equals to 1 and to “valueseqo” OutPort if “Cmd” equals to 2 and wait if the value of “Cmd” is 0.

The JLSRController RTC reads the data from its InPort which is sent by the JLSRA dapter RTC and JLSR robot RTC and transforms them to current torque values through a PD controller as its core logic, as shown in the equation (1). The JLSRController RTC then output the results to JLSR robot RTC for simulation:

\[
\tau_p(q) = -D(q) \ddot{q} + P(q, \dot{q})
\]  

(1)
where, $P \in \mathbb{R}^{n \times n}$ and $D \in \mathbb{R}^{n \times n}$ are the P/D gain parameters, $q_{\text{ref}} \in \mathbb{R}^n$ is the reference (desired) joint values, $q_{\text{cur}} \in \mathbb{R}^n$ is the current joint values and $q_{\text{ref}}^/ \in \mathbb{R}^n / q_{\text{cur}}^/ \in \mathbb{R}^n$ is the time derivative of $q_{\text{ref}}$ / $q_{\text{cur}}$
.

The transition process of JLSR controller RTC is shown in Fig. 8. It was developed based on RT middleware (Open RTM-aist 1.0.0) (Ando et al., 2005). The transition process includes:

- **Rtc-starting-entry**: JLSRController initialization read in PD gain values and the torque limit value.
- **Rtc-stopping-entry**: stop reading the InPorts of this RTC, calculating the torque values and writing to the OutPorts.
- **Rtc-active-do**: read in the values from JLSRAdapter RTC, calculate the output torque values and write to the OutPorts of this RTC.

We distributed the simulation system to be executed on two computers including a notebook with Linux Ubuntu 10.04 (2.20GHz, Core2, 4GB RAM) and a Dell desktop with Window 7 Professional (3.10GHz, Core i5, 4GB RAM). The notebook’s IP is 192.168.1.101, while the desktop PC’s IP is 10.0.0.113. The created RT system and other corresponding CORBA servers can be seen in Fig. 9.

After the setup of this distributed simulation system, the simulation result can be seen in Fig. 5. And the action of the picking task for the real robot could be seen in Fig. 10. The procedure of controlling the real robot is almost the same as the simulation.

**CONCLUSION**

Using RT-Middleware technology, the development of robotic systems could be divided into two parts, namely, component development and system integration. These two parts can be carried out independently, thus achieving a scientific division of labor: the component developer can concentrate on implementing the main logic and the integrator can
concentrate on the design of the whole system without thinking about the detail of the implementations. We develop robotic functional elements as RT Components for the guide robot, which make it easier to create the distributed robot simulation system by re-using existing modules and lower the cost of development of new robot application.

From the application of the distributed simulation system, we could also see that developing the software modules into RT Components and using RTM to integrate them into a robot system can achieve the reusability and interoperability of the modular components and make the robot system be network distributed easily. In the meanwhile, the distributed simulation system could be transferred and applied to the real robot system easily and quickly. So it can greatly improve development efficiency and adapt to rapid changes in demand.

ACKNOWLEDGMENT

This study was supported in part by the National High Technology Research and Development Program of China under grant 2007AA041602 and 2011AA040801; the authors gratefully acknowledge the support from YASKAWA Electric Cooperation for supporting the collaborative research funds and address our thanks to Mr. Ikuo Nagamatsu and Mr. Masaru Adachi at YASKAWA for their cooperation.

REFERENCES


